U. S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
NATIONAL METEOROLOGICAL CENTER

OFFICE NOTE 148

NMC's Rotating Partitioned Dataset Scheme

Peter Chase
Automation Division

JUNE 1977

NMC's Rotating Partitioned Dataset Scheme

Peter Chase
Automation Division
June 1977

Why a rotating partitioned dataset?

It may happen that one program is reading the file that a second program is simultaneously rewriting.

This is known as a race condition, because the results depend upon which program wins the race.

In IBM's Operating System (OS) there are techniques for preventing interference among programs running on the same machine. However, IBM's Asymmetric Multiprocessing System (ASP) has no provision whatsoever for preventing interference among programs running on different machines in the same system.

The rotating partitioned dataset scheme solves the interference problem in the case where there is only one program at a time which rewrites files. It also solves the problem of avoiding compressing the partitioned dataset (PDS) when it runs out of space.

The scheme described below could have been implemented using several datasets instead of one PDS by modifying the disk volume table of contents (VTOC) instead of the PDS directory. However, it was felt safer to leave the VTOC alone.

How does it work?

The PDS is initialized with a fixed set of membernames and a set of interchangeable data spaces. There may well be more membernames than spaces: the extra membernames point to a special empty data space.

There is a particular membername called NEXT which never points to an empty data space. The updating programs always write in NEXT. Following this, the updating programs swap data spaces around among the membernames so that the following results are achieved:

(1) A current name (call it curnam) now points to the rewritten data space previously pointed to by NEXT.

(2) A previous name (call it prvnam) now points to the data space previously pointed to by curnam.

(3) NEXT now points to some re-useable space.

(4) A membername CURRENT is an alias for curnam.

If a retrieving program was in the process of reading curnam, it can continue to do so since the associated data space has not been over-written. It has only been renamed prvnam.

If a retrieving program had not opened its input file to read curnam at the time the space swap was performed, it will now read the new contents of curnam.

There is never a dangerous or ambiguous period in this scheme, for the following reasons:

(1) No membername entry in the PDS directory extends from one directory block to the next, so that all of the entry gets simultaneously rewritten.

(2) The membername entries stay in fixed locations (actually in alphabetical order) in the directory. Therefore there is no time during the process of updating the directory when a membername entry would appear other than exactly once.

The retrieving programs can all read the data without special software in the usual fashion:

//ddname DD DSN=fully.qualified.dsname(member),DISP=SHR

Permanent and temporary membernames

Certain membernames may be flagged permanent. This means that their data spaces will never be taken away from them until they are updated. They will never point to the empty space.

The remaining membernames are temporary. This means that in time their data spaces will be taken from them to be re-used. Following this these membernames will point to the empty space, which contains only an 'ENDOF FILE' block.

The method for choosing which of the temporary membernames will have its data space taken away is the least-recently-used (LRU) algorithm. Associated with each temporary membername is an age. The age is initially zero for a current name and one for a previous name when that membername is updated. Thereafter, each time the directory is updated, the age is incremented for all temporary non-empty membernames. If space is needed, the membername with the first gratest age is chosen.

With this method, a data space should remain accessible to any retrieving program until the second subsequent directory update.

The membernames to be flagged permanent are chosen at the time the rotating PDS is initialized and never change thereafter.

Rewriting a membername

The updating program rewrites a membername curnam in the following fashion:

(1) Using routine W3AK27, he writes all of his blocks of data in the space pointed to by membername NEXT.

(2) If the end-of-data return has not been reached, he writes 'ENDOF FILE' blocks until it is reached.

(3) He calls W3AK28 to point curnam to this space, to locate re-useable space for NEXT, and to point prvnam to the space previously pointed to by curnam.

The process of rewriting in place requires that the blocksize must be the same for all physical blocks at all times. Therefore W3AK27 obtains the blocksize from the data control block after the file has been opened and computes the logical record length and data length from the blocksize. If a variable format is used, the block and record descriptor words will be constructed by W3AK27. Thus the length of the data arrays is fixed and cannot span more than one record even though the record format is nominally VS.

The directory structure

The directory of a PDS is actually a separate file (as is each member of a PDS) terminated with an end-of-data block. This directory file has an organization entirely distinct from the organization of the member files:

-- It is a direct access file.

-- It consists of 256-byte blocks, exactly as many blocks as are specified in the third coordinate of the SPACE parameter on the DD statement that created the PDS.

-- Each block is written with an 8-byte hardware key which is the same as the last membername in that block. This allows the particular block you need to be quickly located.

-- The first two bytes in each block contain a count of the number of active bytes in the block. The count includes the first two bytes themselves.

-- Entries are of varying size, from 12 bytes up to 76 bytes long. (We don't consider pointers here.) All of an entry is contained on one physical block.

-- Membername entries occur in alphabetical order in the directory. The last entry in the directory is a fence, consisting of 8 hexadecimal FF bytes (all one-bits), and is only 8 bytes long. (The directory, when created, has this fence as the first and only entry.) There may be unused directory blocks past the fence.

-- The first entry in each block begins in byte 3. Each entry has the format shown below.

-- The formula for computing the number of bytes in a given directory entry is as follows:

length = 12 + 2(number of user halfwords)

where the number of user halfwords is in bits 4-8 of the 12th byte of the entry.

PDS Directory Entry Format

Bytes	Contents
1-8	Membername, from 1 to 8 EBCDIC characters, left-justified with right blank fill. Characters may be alphanumeric or national, but the first character may not be numeric.
9-11	TTR, track and record pointer to beginning of member space.
12	Flags and counts, as follows: Bit 1 = 1 if membername is an alias for another name. Bits 2-3 indicate the presence of pointers. We don't use pointers, so these bits will be zero. Bits 4-8 give the number of user halfwords, 0-31. In this scheme, the number will always be 1 or more.
13-14	The first user halfword, used to control the rotating PDS scheme: Bit 1 = 1 if this membername points to the empty space. Bit 2 = 1 if this membername is permanent; i.e., not to be aged. Bits 3-8 are not used. Bits 9-16 are the age of the membername, 0 to 255 max.

Bytes 1-12 are standard. Bytes 13-14 are especially for the rotating PDS scheme. Currently there is only one user halfword and bits 4-8 of byte 12 are 00001.

The software expects to find membernames NEXT and CURRENT in the directory. In addition, it expects a number of pairs of names C<u>xxxxxxx</u>, P<u>xxxxxxx</u>. These are the current and previous names, respectively.

Initialization

Initialization of the rotating PDS is accomplished by a program ROPIN. ROPIN is controlled by input cards, as follows:

Card 1

Columns	Contents
	(All right-justified integers)
1-10	Number of data spaces to be formatted, not including the empty space. There must be one data space for each permanent membername, one data space for NEXT, and one spare.
11-20	Number of physical blocks in each data space. Note that all data spaces will be the same size, except for the empty space which will contain only one 'ENDOF FILE' block.
21-30	Number of bytes of data in each physical block. This count does not include the block or record descriptor words for variable record formats; in this case, the number of bytes of data must be exactly 8 bytes less than the blocksize. Note that all blocks will be the same size.
31-40	Number of directory blocks specified on the SPACE parameter of the DD statement that created the PDS.

Card 1 will cause data spaces to be preformatted using routines W3AK10-W3AK15. The following membername cards will initialize the directory:

Cards 2,...,N (N ≤ 200)

Columns	Contents
1-8	Membername, 1-8 EBCDIC characters, left-justified with right blank fill. Characters may be alpha-numeric or national, but first character may not be numeric. Names need not be in alphabetic order here.
11	'T' if this is a permanent membername; that is, not to be aged. Permanent membernames keep their data spaces until they are updated.

13 'T' if name is to be an alias. (Use once only for CURRENT.)

There should be membername cards for NEXT, CURRENT, and pairs of the form Cxxxxxxx, Pxxxxxxx. For example, suppose that you want permanent membernames for ØØZ, Ø6Z, 12Z and 18Z data and temporary membernames for Ø3Z, Ø9Z, 15Z and 21Z data. You might have the following membername cards:

```
          1 1
1         1 3
NEXT      T F       (Note: NEXT is considered permanent.)
CURRENT   T T
CØØZ      T F       (Note: Here are the permanent data
CØ6Z      T F        membernames.)
C12Z      T F
C18Z      T F
CØ3Z      F F       (Note: Here are the temporary data
CØ9Z      F F        membernames.)
C15Z      F F
C21Z      F F
PØØZ      F F       (Note: For every membername beginning
PØ3Z      F F        with 'C' we have a previous membername
PØ6Z      F F        beginning with 'P'. They are all
PØ9Z      F F        temporary.)
P12Z      F F
P15Z      F F
P18Z      F F
P21Z      F F
```

Initially data spaces will be assigned to NEXT and all the permanent membernames. If there are any remaining unassigned data spaces, they will be assigned to temporary membernames. After all the data spaces have been assigned, any remaining temporary membernames will be pointed to the empty space.

ROPIN may be executed as follows:

```
//     EXEC  PGM=ROPIN
//STEPLIB  DD  whatever
//FTØ1FØØ1 DD (the rotating PDS--it may be created with this DD)
//FTØ6FØØ1 DD SYSOUT=A
//FTØ5FØØ1 DD *
  (control cards)
/*
```

ROPIN uses routines W3AK29 and W3AK3Ø to read and write the directory.

Further information

Further information about these routines may be found in the writeups for W3AK27, W3AK28, W3AK29 and W3AK3Ø.

- - - -